

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO – PUC-RIO
DEPARTAMENTO DE INFORMÁTICA - DI
PROGRAMA DE PÓS GRADUAÇÃO EM INFORMÁTICA
INF2541 – INTRODUÇÃO À COMPUTAÇÃO MÓVEL
PROF. MARKUS ENDLER

MONOGRAFIA:
PLATAFORMAS DE DESENVOLVIMENTO PARA DISPOSITIVOS
MÓVEIS

ALUNA: JULIANA FRANÇA SANTOS AQUINO
MATRÍCULA: 0711302
JULIANA@LAC.INF.PUC-RIO.BR

RIO DE JANEIRO, 10 DE DEZEMBRO DE 2007.

ÍNDICE

1	INTRODUÇÃO	3
2	ANDROID	3
2.1	HISTÓRICO.....	3
2.2	DEFINIÇÃO	4
2.3	ARQUITETURA	4
2.3.1	KERNEL DO LINUX.....	5
2.3.2	BIBLIOTECAS	5
2.3.3	ANDROID RUNTIME	6
2.3.4	FRAMEWORK DE APLICAÇÕES.....	7
2.3.5	APLICAÇÕES	8
2.4	BLOCO DE APLICAÇÕES	8
2.4.1	ACTIVITY.....	8
2.4.2	INTENT RECEPTOR	9
2.4.3	SERVICE	9
2.4.4	CONTENT PROVIDER.....	9
3	QTOPIA	10
3.1	HISTÓRICO.....	10
3.2	ARQUITETURA	10
3.2.1	QTOPIA CORE	11
3.2.2	QTOPIA PLATFORM.....	12
3.2.3	QTOPIA PHONE EDITION	12
3.3	QTOPIA GREENPHONE	12
4	CONSIDERAÇÕES FINAIS	12
	REFERÊNCIAS	14

1 INTRODUÇÃO

Com o grande avanço da tecnologia, os dispositivos móveis estão se tornando mais poderosos com relação às suas capacidades de armazenamento, de processamento e de comunicação, e mais acessíveis aos consumidores, ao mesmo tempo. Hoje, há 1,5 bilhões de televisores no mundo, enquanto que 1 bilhão de pessoas têm acesso à *Internet*, mas quase 3 bilhões de pessoas possuem um dispositivo móvel, tornando-se um dos produtos mais promissores do mundo. Dentre esses dispositivos móveis, estão incluídos *handhelds*, PDAs e telefones celulares.

Os dispositivos móveis oferecem conectividade e poder de uso a qualquer lugar e em qualquer momento, tornando-se importantes, tanto para uso pessoal, quanto profissional.

Com o uso cada vez maior de dispositivos móveis, o número de plataformas e ambientes de desenvolvimento cresceu proporcionalmente.

A escolha de uma plataforma ideal para o desenvolvimento de um projeto significa optar por uma solução que forneça os melhores benefícios, em termos de custos, eficiência e tempo de desenvolvimento esperados para a finalização do projeto.

Sendo assim, a presente monografia tem como objetivo fazer uma explanação sobre duas plataformas de desenvolvimento de aplicações para dispositivos móveis: o Android (ver seção 2), lançado pela *Open Handset Alliance* [1], e o Qtopia (ver seção 3), produto da Trolltech [1].

2 ANDROID

2.1 HISTÓRICO

Em 05 de novembro de 2007, o OHA (*Open Handset Alliance*) anunciou a plataforma Android [3]. *O Open Handset Alliance* é um grupo de mais de 30 empresas que está desenvolvendo essa plataforma. As empresas dessa aliança estão trabalhando juntas para oferecer uma plataforma de desenvolvimento que permita aos desenvolvedores implementarem e estenderem as aplicações dos seus dispositivos móveis. Essa aliança tem como objetivo também lançar *handsets* e serviços usando a plataforma Android no segundo semestre de 2008.

A aliança OHA é composta por um grupo bastante heterogêneo de empresas, que compreende desde operadoras de celular a fabricantes de *handsets*. Dessa aliança, fazem parte:

- Operadoras de celular:



- Companhias de semicondutores:



- Fabricantes de *handset*:



- Companhias de *software*:



2.2 DEFINIÇÃO

Android é uma pilha de *softwares* para dispositivos móveis que inclui um sistema operacional, um *middleware* e um conjunto de aplicações chaves. Os desenvolvedores podem criar aplicações para a plataforma usando o Android SDK (ver seção **Error! Reference source not found.**). As aplicações para essa plataforma são escritas usando a linguagem de programação Java e executam sobre o Dalvik, uma máquina virtual customizada para dispositivos com restrições de recursos, como pouca capacidade computacional, baixa capacidade de armazenamento e baterias com baixo nível de energia.

2.3 ARQUITETURA

O diagrama da Figura 1 mostra os principais componentes do sistema operacional Android.

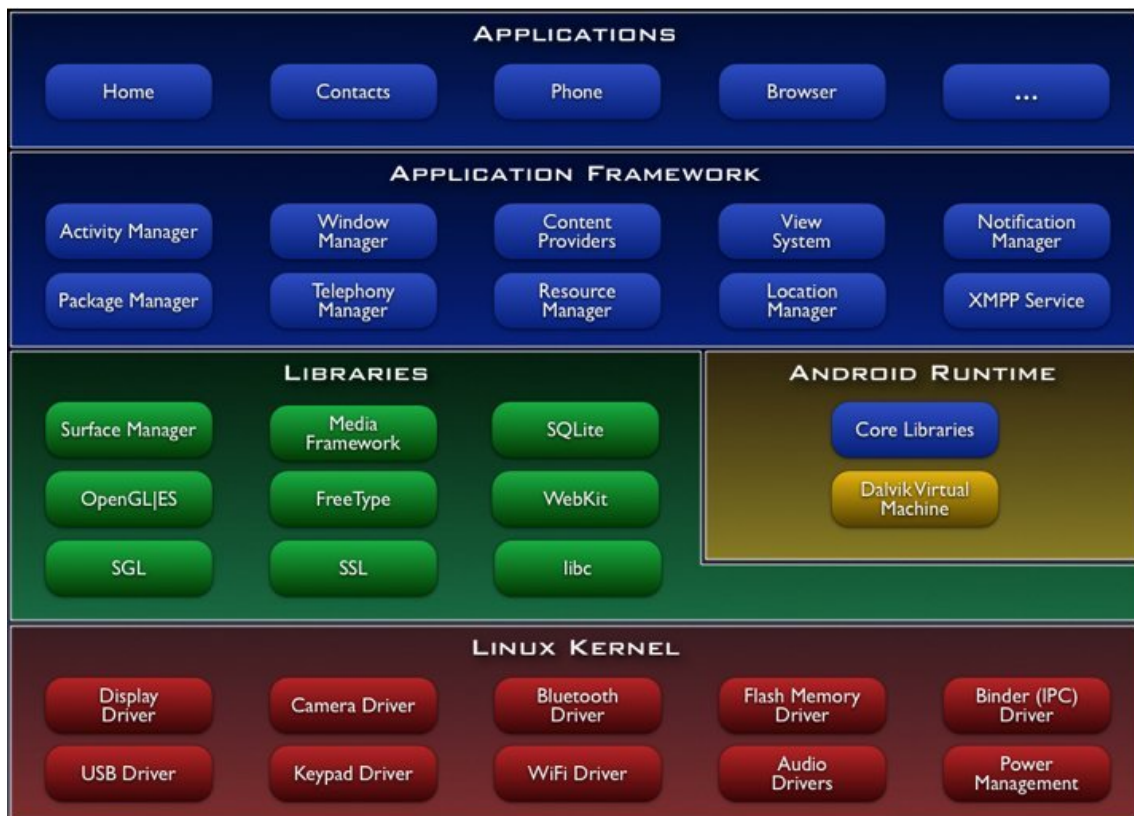


Figura 1 – Arquitetura do Android. Fonte: [4].

Nas seções seguintes, realiza-se uma explicação sobre os módulos da arquitetura Android.

2.3.1 KERNEL DO LINUX

A arquitetura do Android é baseada no *kernel* do GNU/Linux, versão 2.6. O *kernel* do sistema funciona como uma camada de abstração entre o *hardware* e o restante da pilha de *softwares* da plataforma.

O *kernel* GNU/Linux já possui vários recursos necessários para a execução de aplicações, como gerenciamento de memória, gerenciamento de processos, pilha de protocolos de rede, módulo de segurança e vários módulos do núcleo de infra-estrutura. Como o sistema operacional é conhecido, também facilita o surgimento de melhorias aos *drivers* já existentes.

2.3.2 BIBLIOTECAS

O Android inclui um novo conjunto de bibliotecas C/C++ usados pelos vários componentes do sistema. As funcionalidades são expostas através do *framework* do Android (ver seção 2.3.4). Algumas das bibliotecas do núcleo da arquitetura são listadas abaixo:

- *Surface Manager*: Controla e gerencia o acesso ao subsistema de *display*. Compõe transparentemente camadas gráficas 2D e 3D de múltiplas aplicações.

- *3D libraries*: Uma implementação baseada na especificação do OpenGL 1.0.
- *SGL*: Biblioteca usada para compor gráficos 2D.
- *Media libraries*: Essas bibliotecas suportam *playback* e gravação de muitos formatos de áudio e de vídeo, bem como imagens estáticas, incluindo MPEG4, H.264, MP3, AAC, AMR, JPG e PNG.
- *FreeType*: É uma biblioteca usada para renderizar fontes.
- *SSL*: Fornece encriptação de dados enviados pela *Internet*.
- *SQLite*: É uma biblioteca C que implementa um banco de dados SQL embutido. Programas que usam a biblioteca SQLite podem ter acesso a banco de dados SQL sem executar um processo RDBMS em separado. SQLite não é uma biblioteca de cliente usada para conectar com um servidor de banco de dados. SQLite é o servidor. A biblioteca SQLite lê e escreve diretamente para e do arquivo do banco de dados no disco.
- *LibWebCore*: Biblioteca base para o navegador do Android e visões da *web*.
- *System C Library*: Uma implementação do *libc* derivada do BSD.

2.3.3 ANDROID RUNTIME

Toda aplicação Android executa seu próprio processo, com sua própria instância da máquina virtual Dalvik. Dalvik foi escrito de forma que um dispositivo possa executar múltiplas máquinas virtuais concorrentemente de maneira eficiente.

Dalvik executa classes compiladas por um compilador da linguagem Java. Os arquivos *.class* gerados são transformados no formato *.dex* pela ferramenta *dx*, incluída no SDK (*Software Development Kit*) do Android. Esses arquivos *.dex* são executados pelo Dalvik.

A máquina virtual Dalvik também usa o *kernel* do GNU/Linux para prover a funcionalidade de múltiplas *threads* e gerenciamento de memória de baixo nível.

O componente em azul da Figura 2 contém um conjunto de bibliotecas que fornece a maioria das funcionalidades disponíveis no núcleo das bibliotecas da linguagem Java. Isso inclui classes para manipulação de arquivos, entrada e saída, entre outros.



Figura 2 – Android Runtime. Fonte: [4].

Uma das primeiras diferenças de Dalvik para a máquina virtual Java (JVM) é que Dalvik é baseado em registradores e a JVM é baseada em pilhas. A escolha da abordagem baseada em registradores traz um benefício em ambientes restritos, como telefones celulares, e uma análise mais aprofundada concluiu que máquinas virtuais baseadas em registradores executam mais rapidamente os programas do que máquinas baseadas em pilhas.

Outra diferença é que o Dalvik é otimizado para permitir múltiplas instâncias da máquina virtual para executar ao mesmo tempo em memória limitada. E cada aplicação executa como um processo linux separado. Então dado que cada aplicação possui seu próprio processo, isso permite instalação dinâmica, ativação e desativação. Todavia, Dalvik poderia ter usado OSGi [5] para habilitar essas características dentro de um único processo. Processos separados previnem que todas as aplicações sejam fechadas se a máquina virtual deixar de funcionar. Entretanto, OSGi ainda pode ser portado para a plataforma Android.

2.3.4 FRAMEWORK DE APLICAÇÕES

Os desenvolvedores têm acesso completo à mesma API que é usada pelas aplicações *core* da plataforma. A arquitetura da aplicação foi projetada para simplificar o reuso dos componentes. Qualquer componente pode publicar suas capacidades e quaisquer outros componentes podem então fazer uso dessas capacidades, sujeito às restrições de segurança reforçadas pelo *framework*. Esse mesmo mecanismo permite que os componentes sejam substituídos por outros em tempo de desenvolvimento.

O fundamento de todas as aplicações do *framework* é um conjunto de serviços e sistemas (Figura 3), que inclui:



Figura 3 – *Framework* de Aplicações. Fonte: [4].

- *Activity Manager* (Gerenciador de atividade): Gerencia o ciclo de vida das aplicações.
- *Package Manager* (Gerenciador de pacotes): Mantém quais aplicações estão instaladas no dispositivo.
- *Window Manager* (Gerenciador de janelas): Gerencia as janelas das aplicações.
- *Telephony Manager* (Gerenciador de telefonia): Componentes para acesso aos recursos de telefonia.
- *Content Providers* (Provedores de conteúdo): Permitem que as aplicações acessem os dados de outras aplicações (como contatos) ou compartilhem os seus próprios dados.
- *Resource Manager* (Gerenciador de recursos): Fornece acesso a recursos gráficos e arquivos de *layout*.
- *View System* (Visão do sistema): Um conjunto rico e extensível de componentes de interface de usuário. As visões podem ser usadas para construir uma aplicação, elas incluem listas, grids, caixas de texto, botões, dentre outras.
- *Location Manager* (Gerenciador de localização): Gerencia a localização do dispositivo.
- *Notification Manager* (Gerenciador de notificações): Permite que todas as aplicações exibam alertas na barra de status.
- *XMPP Service* (Serviço XMPP): Suporte para uso do protocolo XMPP (*Extensible Messaging and Presence Protocol*) [6].

2.3.5 APLICAÇÕES

Android fornece um conjunto de aplicações básicas, que inclui um cliente de email, um programa SMS, um calendário, mapas, navegador, contatos, entre outras.

2.4 BLOCO DE APLICAÇÕES

Há quatro blocos principais de construção de uma aplicação Android:

- *Activity* (Atividade)
- *Intent Receiver* (Receptor de Intenção)
- *Service* (Serviço)
- *Content Provider* (Provedor de Conteúdo)

Nem toda aplicação necessita ter esses quatro blocos, mas uma aplicação Android deverá ser escrita com um ou mais desses blocos de construção. Uma vez decidido quais componentes são necessários à aplicação, é necessário listá-los em um arquivo chamado *AndroidManifest.xml*. Este arquivo deve conter os componentes da aplicação, além das capacidades e requerimentos para tais.

2.4.1 ACTIVITY

É o mais comum dos quatro blocos de construção Android. Uma *activity* é usualmente uma tela em uma aplicação Android. Cada *activity* é implementada como uma única classe que estende da classe base *Activity*. Essa classe irá mostrar uma interface de usuário composta por *Views* e que responde a eventos.

A maioria das aplicações consiste em múltiplas telas. Por exemplo, uma aplicação de mensagem de texto pode ter uma tela que mostra uma lista de contatos, uma segunda tela para escrever mensagem para um contato e outra tela para rever mensagens antigas ou mudar configurações. Cada uma dessas telas deve ser implementada como uma *activity*. A mudança de uma tela para outra é realizada no início de uma nova *activity*. Em alguns casos, uma *activity* deve retornar um valor para a *activity* prévia. Por exemplo, uma *activity*, que permite que o usuário escolha uma foto, retornaria a foto escolhida para o chamador. Quando uma nova tela abre, a tela chamadora é pausada e posta numa pilha. O usuário pode navegar de volta através das telas abertas anteriormente na pilha.

Android usa uma classe especial chamada *Intent* para realizar a mudança de uma tela para outra. Uma *Intent* descreve o que uma aplicação quer fazer. As duas mais importantes partes da estrutura de dados de uma intenção são a ação e os dados usados para responder a essa ação. Alguns valores típicos de uma ação são MAIN (a porta de entrada de uma atividade), VIEW, PICK e EDIT. Os dados são expressos como uma URI (*Uniform Resource Identifier*). Por exemplo, para ver informações de contato de uma pessoa, é necessário criar uma intenção com a ação VIEW e o conjunto de dados para representar a URI de uma pessoa. Há também uma classe relacionada chamada *IntentFilter*. Enquanto uma intenção é efetivamente uma requisição para se fazer alguma coisa, um filtro de intenção é uma descrição de que intenções uma atividade (ou

intent receiver) pode tratar. Uma *activity* que está apta a exibir informações de contato de uma pessoa deverá publicar um *IntentFilter* que diz que ela sabe como tratar a ação VIEW quando aplicada para representar os dados de uma pessoa. *Activities* publicam seus *IntentFilters* no arquivo *AndroidManifest.xml*.

A navegação entre telas é acompanhada por resolução de intenções. Por exemplo, para navegar para frente, uma atividade chama *startActivity(myIntent)*. O sistema então busca os filtros de intenção para todas as aplicações instaladas e escolhe a atividade cuja intenção casa melhor com esse *myIntent*. A nova *activity* é informada do interesse, que causa o carregamento da nova tela. O processo de resolver intenções acontece em tempo de execução, quando o método *startActivity* é chamado, oferecendo dois benefícios chave:

1. *Activities* podem reusar funcionalidades de outros componentes simplesmente ao fazer uma requisição para outro componente na forma de uma intenção.
2. *Activities* podem ser substituídas a qualquer momento por uma nova *activity* através de um *IntentFilter*.

2.4.2 INTENT RECEPTOR

É usado quando o código em uma aplicação deve executar em reação a um evento externo, por exemplo quando o telefone toca ou quando os dados da rede estão disponíveis. Receptores de intenção não mostram uma interface de usuário, embora eles possam usar o gerenciador de notificação para alertar o usuário quando algum evento de interesse ocorre.

Receptores de intenção são registrados no *AndroidManifest.xml*, mas também é possível registrá-los no código usando *Context.registerReceiver()*. Uma aplicação não precisa estar executando para que seus receptores de intenção sejam chamados. O sistema iniciará a aplicação quando um receptor de intenção é disparado. Aplicações podem também enviar suas próprias intenções por *broadcast* para outras aplicações com *Context.broadcastIntent()*.

2.4.3 SERVICE

Um serviço é um código que está executando em *background* sem uma interface gráfica com o usuário. Um exemplo disso é um tocador de música. No tocador, há uma ou mais *activities* que permitem ao usuário escolher músicas e iniciá-las. Todavia, o tocador não deveria ser tratado por uma atividade porque o usuário espera que a música continue tocando em *background*. O sistema então irá manter o serviço do tocador de música executando até que ele termine.

É possível que uma aplicação se conecte a um serviço para poder iniciá-lo (se ele não estiver executando) com *context.bindservice()*. Uma aplicação pode comunicar-se com um serviço através de uma interface do serviço. Para o serviço de música, ele pode ser usado para pausar uma música ou avançá-la, por exemplo.

2.4.4 CONTENT PROVIDER

As aplicações podem armazenar seus dados em arquivos, como o banco de dados SQLite. Um provedor de conteúdo, todavia, é útil se os dados da aplicação podem ser compartilhados com outras aplicações.

Um provedor de conteúdo é uma classe que implementa um conjunto padrão de métodos para permitir que outras aplicações armazenem e recuperem tipos de dados que são tratados pelo provedor de conteúdo.

3 QTOPIA

Qtopia [8] é uma plataforma C++ da Trolltech [1] para o desenvolvimento de aplicações em dispositivos móveis, como PDAs, telefones celulares e *web pads*. As aplicações construídas com essa plataforma executam sobre sistemas operacionais baseados em GNU/Linux.

A Qtopia não tem rival, até agora, como plataforma de aplicações e de interface de usuário para GNU/Linux, permitindo a criação eficiente de aplicações para dispositivos móveis embutidos.

A *Qtopia 4 Series* é a última geração de produtos da família Qtopia da Trolltech. Ela fornece um ambiente robusto e testado para desenvolvimento, herdado do *framework* líder em aplicações, o Qt. A plataforma compreende o *Qtopia Core* [11], o *Qtopia Platform* [12] e o *Qtopia Phone Edition* [13].

3.1 HISTÓRICO

A Qtopia herda todas as características do Qt [4], o *framework* de aplicações, líder da indústria, da Trolltech. O *framework* Qt C++ é usado por diferentes aplicações comerciais desde 1995. O Qt é usado por companhias e organizações diversas, como: Adobe, Boeing, IBM, Motorola, NASA, Skype, entre outras empresas menores.

Desde 2006, há dezenas de diferentes modelos de telefones celulares e outros dispositivos executando a Qtopia, como dispositivos da Sony, da Motorola, da Philips, da Panasonic, da IBM, entre outros [9]. Qtopia também é usado como plataforma de *software* do Greenphone (ver seção 3.3), uma iniciativa da Trolltech de um celular executando sobre um sistema GNU/Linux.

A Qtopia fornece código fonte completo e, com a documentação disponível, os desenvolvedores podem, facilmente, modificar o Qtopia e integrar outras tecnologias para criar distintos dispositivos.

3.2 ARQUITETURA

A arquitetura do *Qtopia* compreende três componentes primários: *Qtopia Core*, *Qtopia Platform* (um subconjunto do *Qtopia Core*) e *Qtopia Phone Edition* (contendo as capacidades do *Qtopia Core* e do *Qtopia Platform*). Um diagrama de alto nível da arquitetura do Qtopia é mostrado na Figura 4.

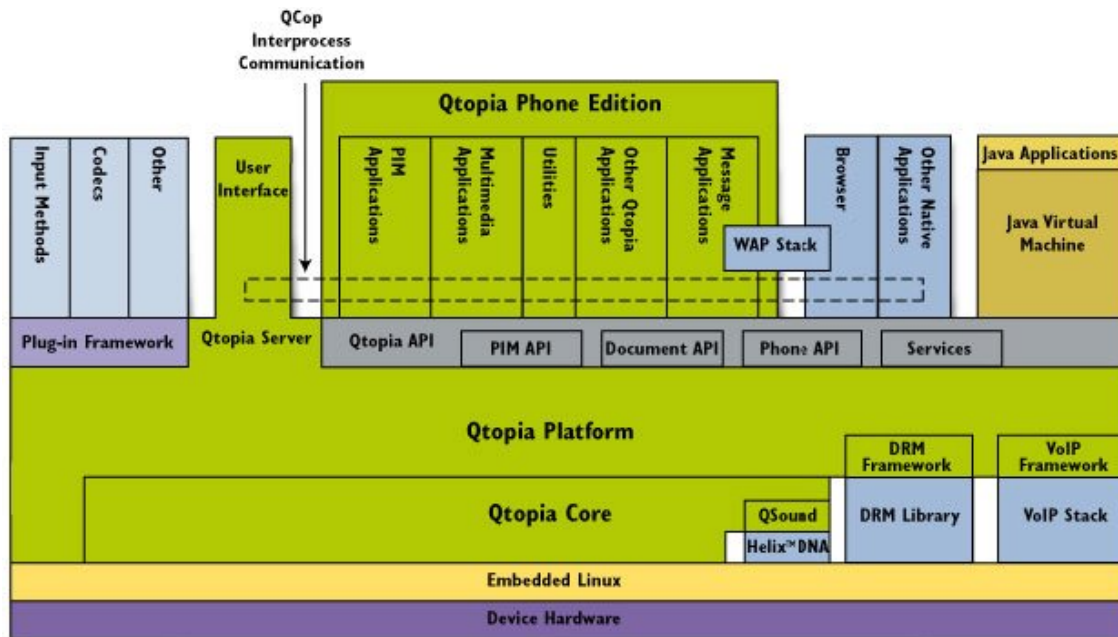


Figura 4 – Arquitetura do Qtopia.

Nas seções seguintes, apresenta-se cada um desses três componentes primários do Qtopia.

3.2.1 QTOPIA CORE

O *Qtopia Core* fornece a base para a família de produtos do Qtopia. Ele fornece o sistema de gerenciamento de janelas, *widgets* básicos e a abstração do sistema operacional do Qtopia.

O *Qtopia Core* é um *framework* C++ para interface com o usuário e desenvolvimento de aplicações para dispositivos embutidos. Ele executa em uma variedade de processadores, como Intel x86, MIPS, ARM e Power PC.

As aplicações do *Qtopia core* usam uma API que escreve diretamente no *framebuffer* ou usa interfaces gráficas especializadas de cada dispositivo. O *Qtopia Core* também inclui muitas ferramentas para o desenvolvimento usando a API padrão do Qt para plataformas *desktops*, além de classes e ferramentas para desenvolvimento e suporte de dispositivos embutidos.

A arquitetura do *Qtopia Core* inclui seu próprio sistema de janelas, ambiente integrado de desenvolvimento, suporte a localização, jogos, multimídia, aplicações PIM (*Personal Information Manager*), integração com Java, suporte para gráficos 2D e 3D e suporte a uma variedade de dispositivos de entrada. O *Qtopia Core* também fornece muitos componentes não-gráficos para tarefas especializadas, como internacionalização, rede, interação de base de dados, classes de *containers*, entrada e saída, XML, interação e uso de *threads*.

As aplicações usando o *Qtopia Core* podem ser escritas com os ambientes de desenvolvimento do Qt. O *Qt Designer* pode ser usado para projetar visualmente as interfaces com o usuário usando o sistema de *layout* do Qt, que automaticamente adapta o espaço da tela disponível. Os desenvolvedores também podem escolher um dos estilos pré-definidos do Qt, ou criar seus próprios estilos.

3.2.2 QTOPIA PLATFORM

O *Qtopia Platform* é construído sobre o *Qtopia Core*, constituindo uma plataforma de aplicações C++ e interface gráfica para produtos eletrônicos baseados em Linux.

Ele oferece um ambiente para aplicações GUI (*Graphical User Interface*) e inclui um conjunto rico em controles, o que fornece funcionalidades padrões de interface com o usuário e tratamento de eventos. Também inclui as bibliotecas de telefonia e a biblioteca PIM. Os processos de telefonia são exemplos de serviços que necessitam estar constantemente disponíveis, isto é, o dispositivo deve sempre estar apto a receber uma chamada. Serviços com estes requisitos devem estar integrados ao *Qtopia Server* (ver Figura 4), que, ao contrário das aplicações, está sempre em execução. Uma máquina virtual Java (JVM) também é integrada. Quando uma aplicação Java é carregada, o *Qtopia Server* carrega a JVM, passando a aplicação Java como parâmetro.

As principais características de telefonia incluem telefonia GSM junto com GPRS e Voz sobre IP (VoIP). O último é integrado durante a interface do usuário junto com GSM, isto é, os contatos incluem um número VoIP.

3.2.3 QTOPIA PHONE EDITION

Qtopia Phone Edition: É construído sobre o *Qtopia Platform* para incluir todo o conjunto de aplicações Qtopia nativas. Isso inclui aplicações PIM, aplicações multimídia e aplicações de mensagens. Ele fornece independência de plataforma e é aprimorado com aplicações pré-definidas. Sua estrutura modular e ferramentas avançadas fazem com que o desenvolvimento de aplicações se torne mais fácil permitindo aos desenvolvedores contruir aplicações para telefones móveis.

3.3 QTOPIA GREENPHONE

O Qtopia Greenphone [10] é um celular baseado na plataforma GNU/Linux, aberto para a criação de novas aplicações. O Qtopia, oferecido como parte do Greenphone SDK, é uma plataforma aberta para o desenvolvimento de aplicativos. O Greenphone é um aparelho GSM/GPRS que promove uma plataforma para a criação de soluções embutidas.

Usando o *Qtopia Phone Edition*, é permitido ao usuário criar seus próprios programas de forma mais rápida.

4 CONSIDERAÇÕES FINAIS

As plataformas Android e o Qtopia são pilhas de *softwares* usadas para desenvolvimento de aplicações para dispositivos móveis que usam um sistema operacional baseado em GNU/Linux. O Android é uma plataforma recente, onde ainda não existem dispositivos que utilizem esta plataforma, enquanto que o Qtopia já é uma plataforma consolidada, usada por vários dispositivos do mercado [9].

A plataforma Android surge como a principal concorrente da Qtopia, principalmente pela força das companhias que estão apoiando o desenvolvimento desta

plataforma, além da promessa da OHA de que a Android terá código aberto. Porém, dispositivos que utilizam essa plataforma só serão lançados na segunda metade de 2008.

Ambas as plataformas oferecem suporte às principais características e requisitos que uma aplicação para um dispositivo móvel requer: API para telefonia, APIs para interface com o usuário, entre outras. A vantagem de se escolher a plataforma Android advém do fato de que pode se fazer uso mais facilmente de todas as aplicações criadas pela Google, como aplicações com uso de mapas. Além disso, a Android fornece uma abordagem bastante simples para a modelagem de aplicações que venham a utilizar essa plataforma. Porém, o Qtopia já fornece várias APIs e classes que herdam todas as características do *framework* Qt, amplamente usado por aplicações de diversas áreas.

REFERÊNCIAS

1. Open Handset Alliance. <http://www.openhandsetalliance.com/>. Acessado em 26 de novembro de 2007.
2. Trolltech. <http://trolltech.com/>. Acessado em 26 de novembro de 2007.
3. Android. <http://code.google.com/android/>. Acessado em 28 de novembro de 2007.
4. Android Architecture. <http://code.google.com/android/what-is-android.html>. Acessado em 29 de novembro de 2007.
5. OsGI. <http://www.osgi.org/>. Acessado em 30 de novembro de 2007.
6. XMPP. <http://www.xmpp.org/>. Acessado em 30 de novembro de 2007.
7. Qt Framework. <http://trolltech.com/products/qt>. Acessado em 28 de novembro de 2007.
8. Qtopia. <http://trolltech.com/products/qtopia>. Acessado em 27 de novembro de 2007.
9. Qtopia Products. <http://trolltech.com/products/qtopia/qtopiainuse/customers>. Acessado em 28 de novembro de 2007.
10. Greenphone. <http://trolltech.com.br/products/qtopia/greenphone/index.htm>. Acessado em 28 de novembro de 2007.
11. Qtopia Core 4.2 Whitepaper. Trolltech. Novembro, 2006. Disponível em <http://trolltech.com/pdf/Qtopia-Core-42-Whitepaper-A4-web.pdf>. Acessado em 01 de dezembro de 2007.
12. Qtopia Platform 4.2 Whitepaper. Trolltech. Novembro, 2006. Disponível em http://trolltech.com/pdf/Qtopia_Phone_Edition_4_ds_web_A4.pdf. Acessado em 01 de dezembro de 2007.
13. Qtopia Phone Edition 4.2 Whitepaper. Trolltech. Novembro, 2006. Disponível em http://trolltech.com/pdf/Qtopia_Phone_Edition_4_ds_web_A4.pdf. Acessado em 01 de dezembro de 2007.